COS 126	Intro to CS	Fall 2015

Midterm 1 Written Exam

There are eight questions on this exam, weighted as indicated at the bottom of this page. There is one question per lecture, numbered corresponding to the lectures, *not in order of difficulty*. If a question seems difficult to you, skip it and come back to it.

Policies. The exam is closed book, though you are allowed to use a single-page one-sided hand-written cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. You have 50 minutes to complete the test. **This exam is preprocessed by computer.** If you use pencil (and eraser), write darkly. Write all answers inside the designated rectangles. Do not write on corner marks.

This page. *Print your name, NetID; fill in your precept on this page*; write and sign the Honor Code pledge.

Discussing this exam. As you know, discussing the contents of this exam before solutions have been posted is a serious violation of the Honor Code.

"*I pledge my honor that I have not violated the Honor Code during this examination.*" NONE

Name: **SOLUTION**

NetID:

Room:

Precept:

P01	P01A	P01B	P02	P02A	P02B	P03	P04	P05	P05A	P06	P11	P12	P13	P14
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

P01	T/Th 12:30pm	Dan Leyzberg
P01A	T/Th 12:30pm	Ted Brundage
P01B	T/Th 12:30pm	Doug Clark
P02	T/Th 1:30pm	Jérémie Lumbroso
P02A	T/Th 1:30pm	Doug Clark
P02B	T/Th 1:30pm	Andrea LaPaugh
P03	T/Th 2:30pm	Jérémie Lumbroso
P04	T/Th 7:30pm	Ming-Yee Tsang
P05	W/F 1:30pm	Stephen Cook
P05A	W/F 1:30pm	Lawrence Lin
P06	W/F 2:30pm	Alan Kaplan
P11	T/Th 1:30pm	Dan Leyzberg
P12	W/F 1:30pm	Donna Gabai
P13	T/Th 3:00pm	Alan Kaplan
P14	W/F 3:00pm	Donna Gabai

Q0. Prologue (8 points). Consider the Boolean `EQ`, defined by the following truth table:

a	b	a EQ b
0	0	1
0	1	0
1	0	0
1	1	1

A. (**3 points**) In the box below, write the bit sequence that results by applying `EQ` in a bitwise fashion to the two given bit sequences.

1 0 1 1 0 0 1 1 0 0 1 1 0 0 1
0 1 0 0 1 1 1 1 0 0 1 1 0 0 0

0 0 0 0 0 0 1 1 1 1 1 1 1 0

B. (**Each question: 1pt if correct, -1pt if incorrect, 0pts if blank**) Indicate whether each of the following statements are True or False, by filling the corresponding bubble.

	True	False
If `a EQ b` is 1 and `b EQ c` is 1, then `a EQ c` is 1.	●	○
`EQ` is equivalent to `NOT XOR`.	●	○
The bitwise `EQ` of a bitstring consisting of all 0s with a random bitstring will have about the same number of 0s as 1s.	●	○
The bitwise `EQ` of *any* bitstring with a random bitstring will have about the same number of 0s as 1s.	●	○
The `EQ` function could be substituted for `XOR` in the encryption/decryption scheme described in lecture, without any ill effects.	●	○

Q1. Types and Casts (9 points). Give the value of each of the following Java expressions. If an expression will not compile or will cause an exception at runtime, put an X under *value*. If the value is a string, enclose it in double quotes. You must fill in every entry. Entries left blank will be marked incorrect.

Expression	Value
`1 / 0`	X
`"800" * 1`	X
`"1" + " - " + "1"`	“1 - 1”
`3.14159 + (int) Math.PI`	6.14159
`1 - 1 - 1 - 1`	-2
`3 / 2.0 + 2 * 5`	11.5
`(8 <= 2) || (2e8 <= 8e2)`	false
`Double.parseDouble("8.5*2")`	X
`"1" + 1 + 1 + "1"`	“1111”

Q2. Loops and Arrays (9 points). Consider the following Java code fragment:

```
1   int[] a = new int[N];
2   a[0] = 1;
3   for (int i = 1; i < N; i++)
4   {
5       int sum = 0;
6       for (int j = 0; j < i; j++)
7           sum = sum + a[j];
8       a[i] = 1 + (2 * sum) / i;
9   }
```

A. (**6 points**) Suppose that `N` is greater than 4. Fill in the following trace of the values of `sum` and `a[i]` *just after* each of the first four iteration of the outer `for` loop.

i	sum	a[i]
1	1	3
2	**4**	**5**
3	**9**	**7**
4	**16**	**9**

B. (**3 points**) In the box below, write *one line of code* that can be substituted for the *four lines of code* in the outer `for` loop above (lines 5 to 8) that would fill the array with the same values.

Valid solutions:

a[i] = 2*i + 1;
a[i] = i + i + 1;
a[i] = a[i-1] + 2;

Q3. More Arrays (9 points).

A. (**3 points**) In the box below, write *one line of Java code* that creates a 4-by-4 (two-dimensional) array named `x` of `double` values and initializes them each to the value `0.0`.

Valid solutions:

```
double[][] x = new double[4][4];
double[][] x = {{0.0, 0.0, 0.0, 0.0}, {0.0, 0.0, 0.0, 0.0},
                {0.0, 0.0, 0.0, 0.0}, {0.0, 0.0, 0.0, 0.0}};
```

B. (**Each question: 1pt if correct, -1pt if incorrect, 0pts if blank**) Indicate whether each of the following statements are True or False, by filling the corresponding bubble.

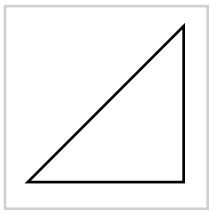
	True	False
Once it has been created, you can change the size of a Java array.	○	●
Once a one-dimensional array has been created and initialized, you can refer to the last element in the array with the code `a[a.length]`.	○	●
Once a one-dimensional array has been created and initialized, you can refer to the first element in the array with the code `a[1]`.	○	●
You can use an array as the return type of a function.	●	○
If you pass an array as an argument to a function, the function can only read the array's entries, not change them.	○	●
If `a` and `b` are arrays of the same length, then the code `a = b` copies each element of `b` to the corresponding in `a`.	○	●

Q4. Input and Output (9 points). Our task is to develop a program `Q4.java` that takes an integer `N` from the command line and then reads `N` point coordinates from standard input and draws the polygon formed by drawing lines connecting successive points, then a line connecting the `N`th point to the first. For example, if the file `Q4ex.txt` contains the values listed at left below, then the command **`java Q4 3 < Q4ex.txt`** should draw a right triangle, and the command **`java Q4 4 < Q4ex.txt`** should draw a square, as depicted at right.

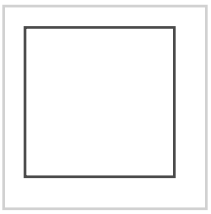
`Q4ex.txt`

```
0.1 0.2
0.8 0.2
0.8 0.9
0.1 0.9
```

`java Q4 3 < Q4ex.txt`



`java Q4 4 < Q4ex.txt`



The code below is a solution to this problem, with four snippets of code missing. Fill in the boxes with the code needed to complete this program.

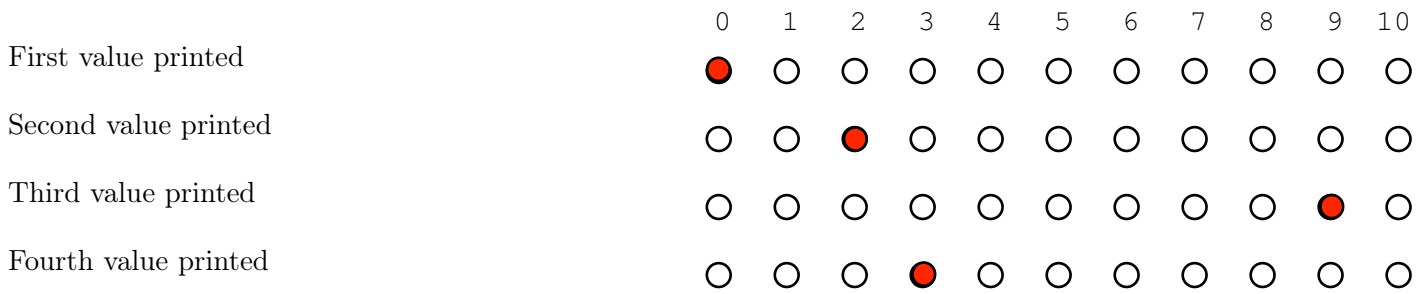
YOU MAY WISH TO SAVE THIS QUESTION FOR LAST—IT MAY BE TIME-CONSUMING.

```
public class Q4
{
   public static void main(String[] args)
   {
      int N = Integer.parseInt(args[0]);
      double xFirst, yFirst, xOld, yOld, xNew, yNew;
      xFirst = StdIn.readDouble();
      yFirst = StdIn.readDouble();
      xOld = xFirst;
      yOld = yFirst;
      for (int i = 1; i < N; i++)
      {
         xNew = StdIn.readDouble();
         yNew = StdIn.readDouble();
         StdDraw.line(xOld, yOld, xNew, yNew);
         xOld = xNew;
         yOld = yNew;
      }
      StdDraw.line(xNew, yNew, xFirst, yFirst);
   }
}
```

Q5. Functions (9 points). Consider the following Java program, which has no purpose other than to test your understanding of functions and scope:

```
public class Q5 {
    public static int f(int x, int[] y, int[] z) {
        x = 1;
        y[0] = x + 1;
        z = new int[4];
        z[3] = 3;
        return z[3];
    }
    public static void main(String[] args) {
        int x = 0;
        int[] y = { 1, 2, 3, 4, 5 };
        int[] z = { 6, 7, 8, 9 };
        z[0] = f(x, y, z);
        System.out.println(x);
        System.out.println(y[0]);
        System.out.println(z[3]);
        System.out.println(z[0]);
    }
}
```

Fill in the requested bubbles.



Q6. Recursion (9 points). Consider the following recursive static method:

```
static int mystery(int a, int b)
{
    if (b == 0) return 0;
    if (a == 0) return mystery(b - 1, a);
    return b + mystery(b, a - 1);
}
```

For each call listed below, write the value returned in the box at the right. (The last response is worth 2 points; the others are 1 point each.)

`mystery(0, 0)`	0
`mystery(10, 0)`	0
`mystery(0, 10)`	0
`mystery(1, 10)`	10
`mystery(10, 1)`	10
`mystery(2, 10)`	20
`mystery(10, 3)`	30
`mystery(200, 300)`	60000

Q7. Performance (8 points). Consider the following tables, which give experimental running times in seconds for four programs `A`, `B`, `C`, and `D` for various values of the input size N.

A

N	*running time*
1,000	21
2,000	80
4,000	325
8,000	1,275

B

N	*running time*
100	2
1,000	25
10,000	260
100,000	2,600

C

N	*running time*
5	1
10	3
20	10
40	100

D

N	*running time*
100	3
200	25
400	200
800	1,599

To the right of each option, fill the bubble of the one-word hypothesis on the order of growth of the running time that best explains the given experimental evidence.

	linear	quadratic	cubic	exponential
A	○	●	○	○
B	●	○	○	○
C	○	○	○	●
D	○	○	●	○